Kenan Khauto
7592047
B.Sc Informatik
kenan.khauto@stud.uni-frankfurt.de

**Bachelor Thesis**

# From Computer Vision to Network Analysis

## Comparing AI Approaches for Understanding Celtic Coin Image Similarities

Kenan Khauto

Submission Date: 19.03.2025

Goethe-Universität Frankfurt am Main
Dr. Karsten Tolle

Bitte dieses Formular zusammen mit der Abschlussarbeit abgeben!

# Erklärung zur Abschlussarbeit

**gemäß § 35, Abs. 16 der Ordnung für den Bachelorstudiengang Informatik vom 17. Juni 2019:**

Hiermit erkläre ich

Khauto, Kenan

*(Nachname, Vorname)*

Die vorliegende Arbeit habe ich selbstständig und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel verfasst.

Ich bestätige außerdem, dass die vorliegende Arbeit nicht, auch nicht auszugsweise, für eine andere Prüfung oder Studienleistung verwendet wurde.

Zudem versichere ich, dass alle eingereichten schriftlichen gebundenen Versionen meiner vorliegenden Bachelorarbeit mit der digital eingereichten elektronischen Version meiner Bachelorarbeit übereinstimmen.

Frankfurt am Main, den 18.03.2025

Unterschrift der/des Studierenden

# Contents

# 1 Introduction

Numismatics, the study of coins and currency, plays a crucial role in understanding the economic, political, and artistic developments of past civilizations (News 2023b). However, one of the biggest challenges numismatists face is the accurate classification and comparison of ancient coins, particularly those from early civilizations such as the Celts. Unlike modern coins with well-documented minting processes, ancient coins often lack standardized inscriptions, making it difficult to categorize them using traditional methods (Exchange 2016).

Traditionally, numismatists rely on expert analysis to compare coins based on their visual attributes, including shape, size, material, and engraving details. This process is highly time-consuming and susceptible to human subjectivity (News 2023a). Additionally, coins that have been worn down over time due to circulation and environmental conditions further complicate classification efforts (Exchange 2016). As a result, there is a pressing need for an automated, objective approach that can accurately analyze coin similarities and group them accordingly.

Artificial Intelligence (AI), particularly Computer Vision, offers a transformative solution to this problem (CoinsWeekly 2023). By leveraging deep learning techniques, AI can extract meaningful features from coin images, allowing for more precise similarity detection (Hinh 2023). This study proposes a multi-step AI-driven framework to aid numismatists in the classification and analysis of ancient coins.

## 1.1 The Role of AI in Numismatics: Addressing Challenges in Coin Classification

### 1.1.1 Challenges in Traditional Numismatic Classification

Numismatics relies heavily on expert knowledge to classify, compare, and analyze coins based on various features such as design, inscriptions, weight, and metal composition. However, this traditional approach has several limitations that make it inefficient and sometimes inconsistent.

#### Time-Consuming Manual Classification

- Numismatic classification typically involves historians and coin experts manually inspecting each coin and identifying key features. This can take a considerable amount of time, especially when dealing with large collections (e.g., museum archives, archaeological finds).
- The process involves cataloging coins based on their attributes (e.g., shape, weight, mint marks, wear patterns), which requires extensive expertise and experience.

- With thousands of coins in circulation from different time periods and regions, manual classification becomes a bottleneck in research and digitization efforts.

#### Subjectivity in Expert Analysis

- Coin classification is not always objective; different experts may categorize the same coin differently based on their interpretations.
- Subtle differences between coins, such as small variations in engravings or damage due to wear, may lead to inconsistent conclusions.
- There is also the problem of bias—some experts might prioritize certain historical factors or aesthetics over technical classifications.
- Disagreements between experts can make it difficult to establish universally accepted classifications, leading to fragmented and sometimes conflicting datasets.

#### Issues with Degraded Coins

- Many ancient coins are heavily worn, corroded, or damaged due to age, environmental exposure, or prolonged use in trade.
- Erosion can make inscriptions unreadable, which is a major issue for identification.
- Some coins are partially broken, missing key features necessary for classification.
- Traditional numismatic methods struggle to handle these cases, as manual identification relies on visible features.
- Reconstruction efforts can be speculative, relying on known historical data and similar coin designs.

### 1.1.2 How AI Enhances Coin Classification and Similarity Detection

Artificial Intelligence, particularly Computer Vision and Machine Learning, provides solutions to these challenges by automating classification, reducing subjectivity, and improving analysis of degraded coins.

#### Machine Learning Approaches for Feature Extraction

Machine learning models can automatically learn meaningful representations of coins, allowing for more precise and scalable classification.

### A. Traditional Image Processing Approaches

- Early methods relied on handcrafted features such as:
  - Edge detection (Sobel Gonzalez and Woods 2002, Canny Canny 1986) to identify coin contours.
  - Histogram of Oriented Gradients (HOG) to detect texture and shapes (Dalal and Triggs 2005).
  - Scale-Invariant Feature Transform (SIFT) for keypoint matching between coins (Lowe 2004).
- While these techniques provided some level of automation, they lacked robustness against variations in lighting, orientation, and wear.

### B. Deep Learning-Based Feature Extraction

- **Convolutional Neural Networks (CNNs)** revolutionized feature extraction (LeCun et al. 1998) by learning hierarchical patterns in images.
- Instead of relying on predefined rules, CNNs automatically extract:
  - Low-level features (edges, textures).
  - Mid-level features (shapes, engravings).
  - High-level features (mint marks, portraits, inscriptions).
- Popular architectures include:
  - **ResNet, EfficientNet, VGG-16** for feature learning (He et al. 2016; Tan and Le 2019; Simonyan and Zisserman 2015).
  - **Vision Transformers (ViTs)** for global pattern recognition (Dosovitskiy et al. 2021).
- These deep models significantly outperform traditional handcrafted methods, providing robust, scalable, and precise feature extraction.

### C. Self-Supervised Learning for Coin Analysis

- **SimCLR, BYOL, and MoCo v3** use contrastive learning to train models without labeled data (T. Chen et al. 2020; Grill et al. 2020; X. Chen, Xie, and He 2021).
- These methods are particularly useful for numismatics, where labeled datasets are scarce.
- The model learns to differentiate coins based on similarity, making it ideal for grouping similar coins even without prior knowledge.

**Code and Reproducibility** To ensure the reproducibility of our experiments and provide an in-depth look at the implementation, all code related to this study, including dataset processing, feature extraction, multimodal fusion, and network analysis, has been made publicly available on GitHub. The repository can be accessed at:

https://github.com/KenanKhauto/AI-driven-coin-analysis

# 2 Dataset Overview: Source, Composition, and Preprocessing

## 2.1 Source and Composition

This study is based on a dataset of numismatic images and metadata, which was kindly provided by Markus Möller, a master's student at the University of Leipzig. The dataset consists of coin images along with weight, diameter, and other relevant details. These details were compiled by Markus from various numismatic collections, where the original information was typically recorded on small paper notes accompanying the coins. Some of these notes exist in printed form as well.

The images originate from multiple museum and private collections, which are summarized in 2.1. The dataset does not originate from a single unified publication but was instead curated from individual coin collection records.

Table 2.1: Summary of Numismatic Collections and Coin Counts

| **Collection Name** | **Number of Coins** |
|---|---|
| Linz Neubau | 166 |
| Archäologische Staatssammlung München | 198 |
| Deggendorf, Kreisarchäologie | 4 |
| Gäubodenmuseum Straubing | 5 |
| Historisches Museum Regensburg | 68 |
| Oberösterreichisches Landesmuseum | 61 |
| Privatbesitz | 43 |
| Staatliche Münzsammlung München | 176 |
| Stadtmuseum Bad Reichenhall | 43 |
| **Total** | **764** |

However, one challenge with the dataset is that the images exhibit inconsistent resolutions and aspect ratios, likely due to variations in imaging conditions, camera setups, and cropping differences. This inconsistency may introduce challenges in feature extraction and similarity computation, requiring preprocessing steps such as image resizing, normalization, and aspect ratio standardization to ensure uniformity in further analysis.

## 2.2 Dataset Preprocessing

Before applying AI-based analysis, a structured preprocessing pipeline is implemented to ensure consistency and quality in the dataset. The preprocessing steps include image standardization, pairing of obverse and reverse images, and normalization. The dataset is loaded

(a) Obverse of Coin 1

(b) Reverse of Coin 1

(c) Obverse of Coin 2

(d) Reverse of Coin 2

Figure 2.1: Obverse and Reverse Views of Two Celtic Coins

using a custom `CoinDataset` class, which efficiently manages the retrieval and transformation of images.

### 2.2.1 Handling Image Size and Resolution Inconsistencies

One of the key challenges in the dataset is that the images exhibit varying resolutions and aspect ratios. To address this, each image is resized to a fixed dimension of 224×224 pixels to maintain uniform input size for AI models. This ensures that feature extraction remains consistent across all images. Resizing is implemented using the following transformation:

```
transforms.Resize((224, 224))
```

### 2.2.2 Image Normalization for Neural Networks

Raw images contain intensity variations due to lighting conditions, camera settings, and background differences. To mitigate these issues, normalization is applied:

- Images are converted to tensors for deep learning processing.
- Each pixel value is normalized using the ImageNet mean and standard deviation:

```
transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
```

This standardization improves model convergence and stabilizes feature extraction (Krizhevsky, Sutskever, and G. E. Hinton 2012).

### 2.2.3 Loading and Organizing the Dataset

The dataset is structured into separate folders for obverse and reverse images. The `CoinDataset` class:

- Extracts image paths and assigns them labels (`"obverse"` or `"reverse"`).
- Filters valid image formats (JPEG, PNG, BMP).
- Implements error handling to ensure robustness.

### 2.2.4 Pairing Obverse and Reverse Coin Images

Since each coin has two corresponding images (obverse and reverse), a pairing mechanism is used:

- Filenames are parsed to extract a base name (e.g., removing suffixes such as _obv or _rev).
- Images are mapped into dictionaries for quick lookup.
- Only coins with a matching obverse and reverse image are included in the paired dataset.

The pairing process is handled using:

```
obverse_dict = {self._extract_base_name(img['filename']): img for img
    in self.obverse_images}
reverse_dict = {self._extract_base_name(img['filename']): img for img
    in self.reverse_images}
self.paired_images = [
        {"obverse": obverse_dict[key], "reverse": reverse_dict[key]}
        for key in obverse_dict.keys() & reverse_dict.keys()
    ]
```

This step ensures that each coin is accurately represented as a two-sided entity, which is essential for similarity-based comparisons.

### 2.2.5 Data Retrieval and Batching

The dataset is designed to return images in different modes:

- **Single Image Mode**: Returns individual images (either obverse or reverse).
- **Paired Mode**: Returns obverse and reverse images together for joint analysis.

The dataset supports PyTorch's `DataLoader`, allowing batch-wise processing for efficient training. Example retrieval:

```
obverse_image, reverse_image, filename_obv, filename_rev = dataset[idx]
```

### 2.2.6 Summary of Preprocessing Steps

| Step | Purpose |
|---|---|
| Resizing (224×224) | Standardizes image dimensions |
| Normalization | Ensures uniform pixel value distribution |
| Loading and Filtering | Organizes images into structured datasets |
| Pairing | Matches obverse and reverse images |
| Data Retrieval | Returns images in required format |

Table 2.2: Preprocessing steps applied to the dataset.

These steps ensure that the dataset is structured, optimized for AI-based processing, and ready for similarity computations and graph-based analysis.

# 3 Feature Extraction and Multimodal Embeddings

## 3.1 Feature Extraction Using Pretrained ResNet-50

### 3.1.1 Introduction to Feature Extraction

Feature extraction is a crucial step in computer vision tasks, where a deep learning model converts raw image data into a meaningful numerical representation (embedding). These feature representations allow efficient comparison and clustering of images based on similarity. In this study, we utilize a **pretrained ResNet-50 model** to extract high-dimensional feature vectors from the obverse and reverse images of Celtic coins.

ResNet-50, a deep residual convolutional neural network, has demonstrated **state-of-the-art performance** in various computer vision applications (He et al. 2016). By leveraging a model pre-trained on the **ImageNet dataset** (Deng et al. 2009), we can benefit from features learned from millions of images, enabling **better generalization** even on historical numismatic images.

### 3.1.2 Feature Extraction Pipeline

The feature extraction process consists of the following steps:

1. **Load a Pretrained ResNet-50 Model**: The final fully connected (FC) layer is removed, keeping only the convolutional feature extractor.
2. **Process Each Coin Image (Obverse/Reverse)**: Images are passed through the network to generate embeddings.
3. **Extract and Store Features**: The extracted feature vectors are saved for later similarity computation.

#### ResNet-50 as a Feature Extractor

ResNet-50 is a **50-layer deep convolutional network** with **residual connections**, designed to mitigate the vanishing gradient problem in deep learning (He et al. 2016). Instead of training a network from scratch, we **remove the classification layer** and use the network as a **fixed feature extractor**. Given an input image $I$, the model outputs a **2048-dimensional feature vector** $F(I)$:

$$F(I) = \text{ResNet-50}(I) \in \mathbb{R}^{2048} \tag{3.1}$$

In the implementation, the following code initializes the ResNet-50 model and removes the final layer:

```
model = models.resnet50(pretrained=True)
self.feature_extractor = torch.nn.Sequential(*list(model.children())[:-1])
      # Remove FC layer
```

**Why ResNet-50?**

- **Trained on ImageNet (1.2M images, 1,000 classes)** → Extracts **generalizable image features**.
- **Removes need for training from scratch**, reducing computational cost.
- **Well-optimized architecture** with residual connections improves gradient flow.

### Feature Extraction Process

To ensure efficient feature extraction, the dataset is processed in mini-batches using PyTorch's `DataLoader`. Each batch of images is:

1. **Loaded into the model** (with `torch.no_grad()` to disable gradient computation).
2. **Passed through the feature extractor**, producing a **2048-dimensional feature vector** per image.
3. **Stored in .npy files** for later retrieval.

The full extraction pipeline is implemented as follows:

```
with torch.no_grad():
    for data in dataloader:
        images, batch_filenames = data
        images = images.to(self.device)

        # Extract features
        batch_features = self.feature_extractor(images).squeeze(-1).
            squeeze(-1)

        # Store results
        features.extend(batch_features.tolist())
        filenames.extend(batch_filenames)
```

The extracted features are then **saved as NumPy arrays (`features.npy`)**, while filenames are stored separately in text files (`filenames.txt`). This allows efficient retrieval without recomputing the embeddings.

**Standardizing Extracted Features:** Feature standardization is a widely used preprocessing step in machine learning (Bishop 2006). In this study, we use `StandardScaler`, a well-established normalization technique from the `scikit-learn` library (Pedregosa et al. 2011). This method ensures that feature distributions are centered at zero with unit variance, improving numerical stability.

After extracting ResNet-50 embeddings, the feature values are standardized using `StandardScaler` to ensure zero mean and unit variance. The transformation is applied as follows:

```
from sklearn.preprocessing import StandardScaler

# Load feature vectors (excluding filenames)
features_ = data.iloc[:, 1:].values

# Apply StandardScaler
scaler = StandardScaler()
features_scaled = scaler.fit_transform(features_)
```

**Mathematical Definition of Standardization:** Feature standardization ensures that each feature has a mean of zero and a standard deviation of one. Given a dataset with $n$ samples and $d$ features, each feature $x_j$ is standardized using:

$$\tilde{x}_j = \frac{x_j - \mu_j}{\sigma_j} \tag{3.2}$$

where:

- $\mu_j = \frac{1}{n} \sum_{i=1}^{n} x_{i,j}$ is the mean of feature $j$.
- $\sigma_j = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (x_{i,j} - \mu_j)^2}$ is the standard deviation.
- $\tilde{x}_j$ is the transformed (standardized) feature value.

This transformation ensures that all features are on a comparable scale, preventing high-magnitude features from dominating distance-based similarity calculations. It is particularly useful when applying clustering algorithms or computing similarity scores based on Euclidean distances.

### Handling Paired Images (Obverse and Reverse)

Since each coin has **two corresponding images** (obverse and reverse), the dataset allows feature extraction for:

- **Individual images** (obverse or reverse alone).
- **Paired images** (extracting features for both obverse and reverse in a single pass).

For paired images, the extraction pipeline processes **both images simultaneously**, ensuring that the corresponding obverse and reverse embeddings are stored together:

```
if self.label == "paired":
    obverse_features = self.feature_extractor(obverse_images).squeeze(-1).
        squeeze(-1)
    reverse_features = self.feature_extractor(reverse_images).squeeze(-1).
        squeeze(-1)
    features.extend(obverse_features.tolist() + reverse_features.tolist())
    filenames.extend(obverse_filenames + reverse_filenames)
```

This ensures that similarity computations can consider both views of the coin.

### 3.1.3 Visualization of Feature Extraction

To better understand the structure of the extracted feature embeddings, we apply t-Distributed Stochastic Neighbor Embedding (t-SNE) Maaten and G. Hinton 2008 to visualize high-dimensional features in a 2D space.

t-SNE is particularly useful for observing local similarities between data points. If the extracted features are meaningful, we expect coins with similar visual structures to form clusters in the 2D projection.

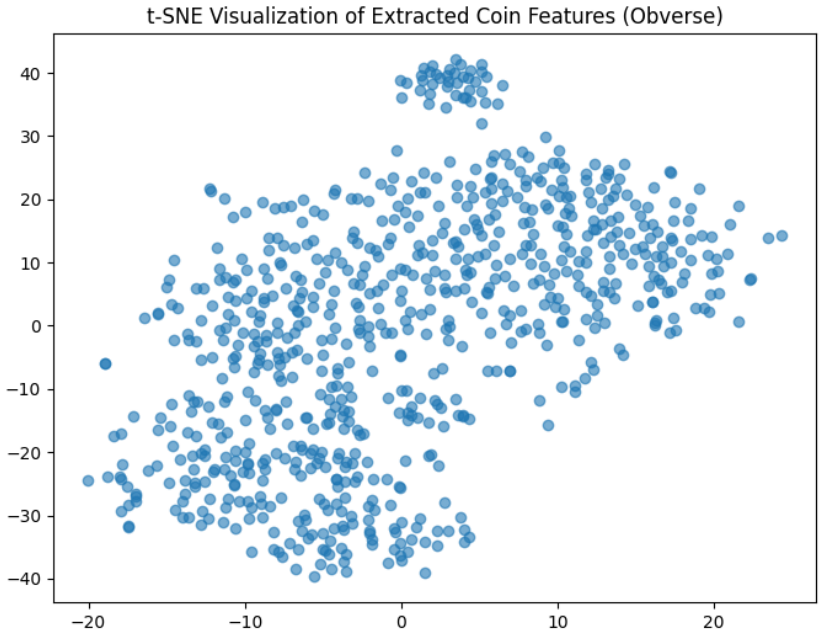


Figure 3.1: t-SNE visualization of RGB-based obverse coin features.

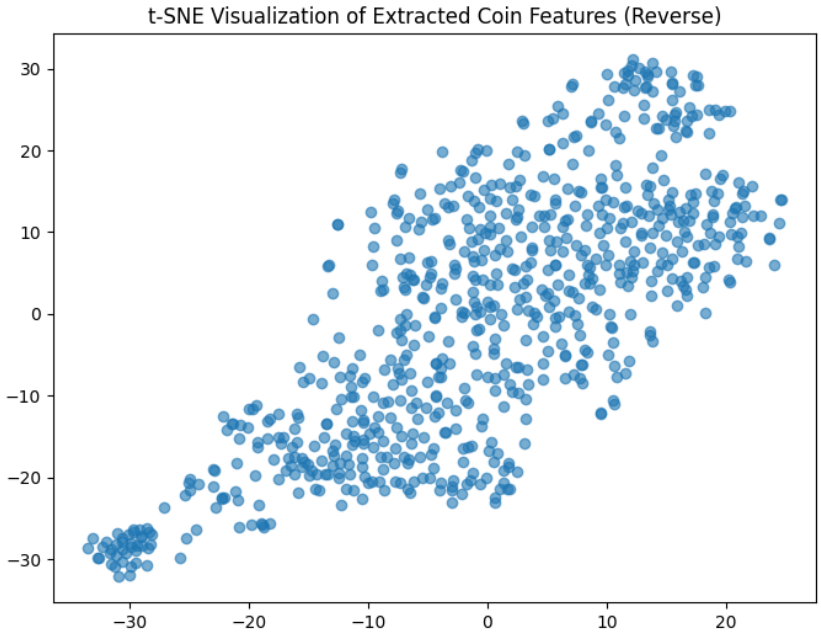


Figure 3.2: t-SNE visualization of RGB-based reverse coin features.

**t-SNE Visualization for RGB Features** The RGB-based feature embeddings show a widely spread structure. Notably, the reverse side exhibits a more continuous distribution, suggesting that these features capture a greater degree of similarity compared to the obverse side.

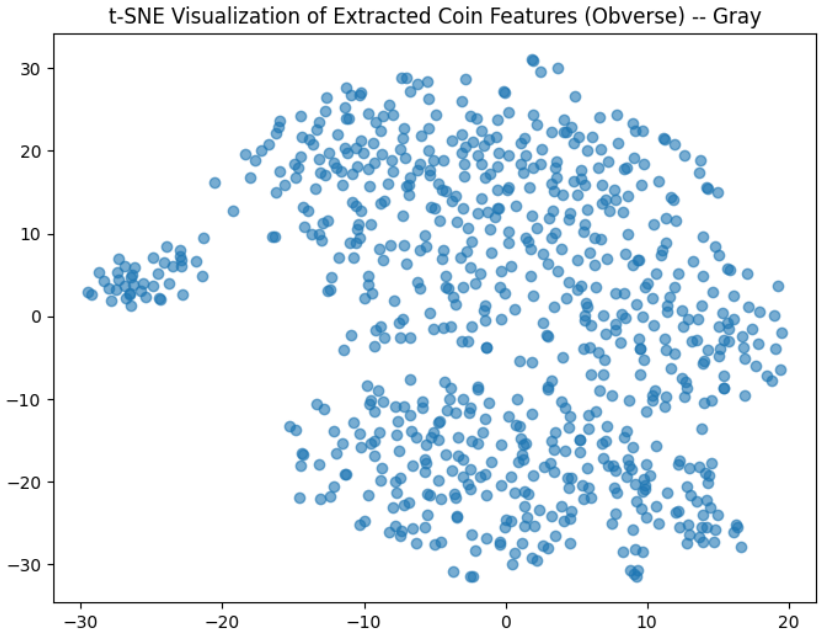


Figure 3.3: t-SNE visualization of grayscale-based obverse coin features.

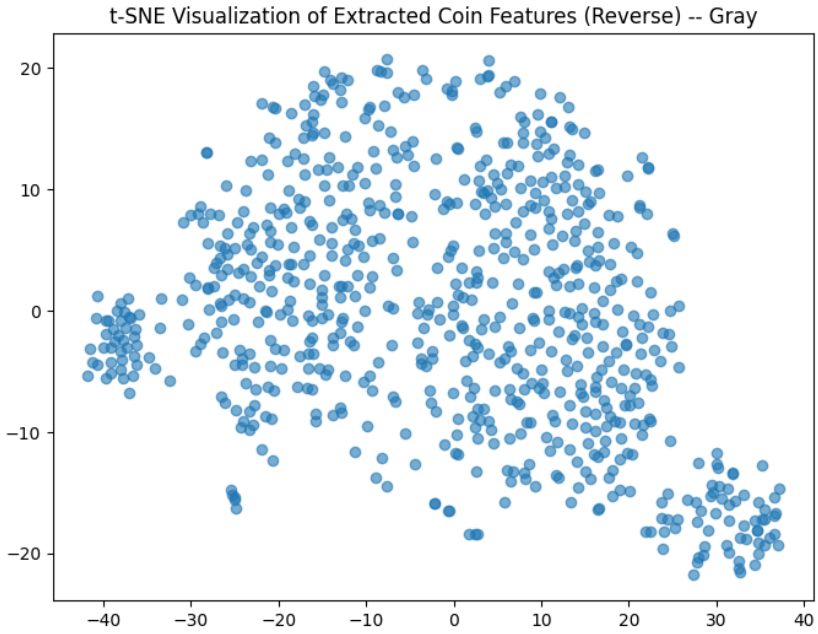


Figure 3.4: t-SNE visualization of grayscale-based reverse coin features.

**t-SNE Visualization for Grayscale Features** Removing color information results in a more compact structure in the t-SNE visualization. This suggests that color introduces variability that is not necessarily useful for similarity detection.

**Comparison and Implications** The comparison between RGB and grayscale feature distributions is summarized in Table 3.1.

| Feature | RGB t-SNE | Grayscale t-SNE |
|---|---|---|
| Cluster Separation | More dispersed | More compact |
| Feature Diversity | Higher due to color differences | Lower, focused on structure |
| Obverse Distribution | Wider spread | More structured |
| Reverse Distribution | Semi-structured | Highly structured |

Table 3.1: Comparison of RGB vs. Grayscale t-SNE Distributions

These results suggest that grayscale embeddings may provide better structured similarity representations, which could improve similarity search and clustering in later stages.

The following Python code generates a t-SNE visualization:

```
from sklearn.manifold import TSNE
import matplotlib.pyplot as plt

features, filenames = extractor.load_features("features.npy", "filenames.
    txt")

# Reduce to 2D using t-SNE
```

```
tsne = TSNE(n_components=2, perplexity=30, random_state=42)
features_2d = tsne.fit_transform(features)

# Plot
plt.figure(figsize=(8, 6))
plt.scatter(features_2d[:, 0], features_2d[:, 1], alpha=0.6)
plt.title("t-SNE Visualization of Extracted Coin Features")
plt.show()
```

### 3.1.4 Summary of Feature Extraction

| Step | Description |
|---|---|
| ResNet-50 Model | Pretrained on ImageNet (final layer removed) |
| Feature Size | 2048-dimensional vector per image |
| Batch Processing | Uses mini-batches for efficient extraction |
| Paired Image Handling | Processes obverse & reverse together |
| Storage Format | Saves embeddings in `.npy` files |

Table 3.2: Summary of feature extraction steps.

## 3.2 Multimodal Feature Extraction: Combining Image and Text

The classification and analysis of ancient coins involve both **visual** and **textual** information. The obverse and reverse sides of coins feature symbols, inscriptions, and patterns that are crucial for identification. In this section, we describe how image and text features are extracted separately and then fused using a **multimodal learning approach**.

### 3.2.1 Processing the Dataset

In a previous chapter, we described the initial preprocessing steps applied to the dataset. However, in this section, we focus on an additional processing step necessary for obtaining the **categories** (types) of the images.

The dataset consists of images stored in a structured folder system. The main dataset folder contains multiple subfolders, where each subfolder represents a specific category or type of coin imagery. The name of each subfolder corresponds to the type of images it contains. Some example categories include:

- **Kopf mit großem Auge** (Head with Large Eye)
- **Magdalensberg**
- **Karlstein**

Each image inside a subfolder inherits the category of that subfolder. However, not all images are stored within these categorized subfolders. Some images remain in the main dataset directory without any assigned category, meaning they are **unclassified**. This presents a challenge, as we need a strategy to handle these unclassified images appropriately.

To address this issue, we introduce a solution in a later subsection where we discuss **CLIP-based text embeddings**(Radford et al. 2021). Specifically, we will define textual templates that allow us to generate meaningful embeddings for both categorized and uncategorized images.

The implementation of this dataset processing logic is integrated into the `CoinDataset` class within the `image_loader.py` file. This class automatically associates each image with its corresponding category if available, while also tracking images that lack classification.

### 3.2.2 Introduction to CLIP

Contrastive Language-Image Pretraining (CLIP) is a **multimodal neural network** developed by OpenAI Radford et al. 2021. It is designed to learn joint representations of **images and text** through contrastive learning, enabling zero-shot learning on a wide range of visual recognition tasks. Unlike traditional supervised learning approaches, which require labeled datasets for each specific task, CLIP generalizes across domains by leveraging large-scale internet data.

#### How CLIP Works

CLIP consists of two separate neural networks:

1. A **Vision Encoder**, typically a convolutional neural network (CNN) such as ResNet-50 or a Vision Transformer (ViT), which converts images into fixed-dimensional feature vectors.

2. A **Text Encoder**, usually a transformer-based model, which encodes textual descriptions into feature vectors of the same dimension as the image embeddings.

During training, CLIP is presented with **image-text pairs**. The model learns to project similar images and textual descriptions closer together in a shared embedding space while pushing dissimilar pairs further apart. This is achieved using a contrastive loss function.

#### Why CLIP is Useful for our Study

Traditional image classification models rely on predefined labels and require fine-tuning for each new dataset. However, CLIP enables a **more flexible and generalizable** approach. Instead of explicitly training a model to recognize specific coin types, we can use text prompts to describe coin features and let CLIP generate meaningful feature representations.

This capability is particularly valuable in numismatics, where:

- Some coins may not be labeled in the dataset (as discussed in 3.2.1).

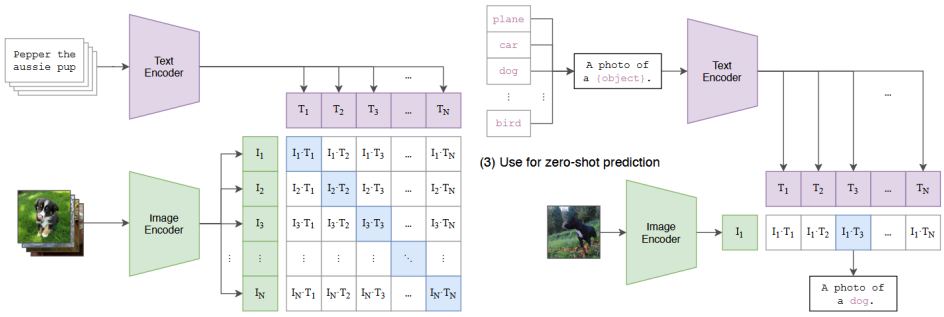


Figure 3.5: Illustration of CLIP's architecture, showing how image and text embeddings are mapped into a shared feature space (Radford et al. 2021).

- Many coin images share visual similarities but differ in textual descriptions.
- There exists a rich historical and semantic context that cannot be fully captured by visual features alone.

By leveraging CLIP, we can extract both image and text embeddings, which we later fuse using a **Multilayer Perceptron (MLP)**. This fusion enhances similarity measurements by integrating multimodal features.

### Zero-Shot Learning with CLIP

One of CLIP's most powerful features is **zero-shot learning**. Because CLIP was trained on large-scale internet data, it can classify images **without requiring labeled training data** for every category. This means that even if a specific coin type was never explicitly trained, CLIP can still recognize similarities based on textual descriptions.

For example, if we describe a coin as "*A Roman coin with a laurel wreath*" and compare it to a database of coin images, CLIP will retrieve the most visually and semantically relevant matches.

### Challenges and Limitations of CLIP

While CLIP provides powerful capabilities, it also has limitations:

- It relies heavily on **pretrained knowledge** and may not recognize rare or highly specialized numismatic features that are not well-represented in internet data.
- The text embeddings' quality depends on **well-crafted textual prompts**.
- CLIP does not perform **fine-grained coin classification** as accurately as models trained on domain-specific datasets.

Despite these challenges, CLIP serves as a **strong foundation** for our multimodal feature extraction approach. In the next subsection, we discuss how we generate textual templates to obtain meaningful text embeddings for coin classification.

### 3.2.3 Creating Text Embeddings Using Templates

In order to leverage the power of CLIP for multimodal representation learning, we need to generate meaningful textual descriptions that align well with the image representations. CLIP maps both images and text into a **shared embedding space**, which means that the quality of the textual descriptions directly impacts how well the textual and visual representations align. Instead of using simple class labels, we construct rich textual prompts to enhance the expressiveness of our text embeddings.

#### The Importance of Text Prompts

The quality of text embeddings in CLIP depends heavily on how well the descriptions capture the **semantic meaning** of the corresponding images. A direct mapping using single-word class labels such as *"Magdalensberg"* or *"Karlstein"* would not fully utilize CLIP's ability to generalize. Instead, template-based descriptions provide contextualized representations that improve feature alignment between modalities.

To generate meaningful textual representations for our dataset, we define structured text templates based on whether the image belongs to the **obverse** or **reverse** side of a coin.

#### Text Template Design

Given that coin images often depict symbolic and historical figures, we introduce structured text templates. The templates dynamically adjust based on the type of coin and the side (obverse or reverse) of the image. The key idea is to create natural language descriptions that provide CLIP with semantic context while maintaining generalizability.

- For coins with known types, we describe them using:

  *"This is the **[side]** of an ancient Celtic coin depicting **[type]**."*

- For coins without assigned types (unknown class), we use a more general template:

  *"The **[side]** of this ancient Celtic coin has an unidentified design."*

By structuring our textual inputs in this way, we ensure that the model learns generalizable textual representations for coins while still allowing fine-grained differentiation.

#### Implementation of Text Templates

The following Python function dynamically generates textual descriptions based on the coin's category and side:

```
def generate_text_description(side, coin_type):
"""
Generates a textual description for CLIP embedding.
:param side: "obverse" or "reverse".
:param coin_type: The type of the coin (or "Unknown").
:return: Formatted text description.
"""
if coin_type.lower() == "unknown":
    return f"The {side} of this ancient Celtic coin has an
       unidentified design."
else:
    return f"This is the {side} of an ancient Celtic coin depicting {
       coin_type}."
```

This function is applied to each sample in the dataset before feeding it into CLIP's **text encoder**. In the next subsection, we describe how we use CLIP to extract both image and text embeddings.

### 3.2.4 Extracting Features with CLIP

After defining meaningful text templates, we extract **multimodal feature embeddings** using OpenAI's CLIP model. CLIP consists of two separate encoders:

- **A Vision Encoder:** A Vision Transformer (ViT-B/32) that converts images into image embeddings.
- **A Text Encoder:** A transformer-based model that converts descriptions into text embeddings.

Both encoders map their respective inputs into a **shared embedding space**, enabling cross-modal comparisons between images and text.

#### Embedding Extraction Pipeline

The CLIP model processes batches of images and text descriptions simultaneously, computing embeddings for each. The full extraction pipeline is as follows:

1. Load the dataset containing obverse and reverse coin images.
2. Preprocess the images using CLIP's built-in image preprocessing pipeline.
3. Tokenize the dynamically generated text prompts using CLIP's text tokenizer.
4. Pass both the images and the text prompts through the corresponding CLIP encoders.
5. Store the extracted embeddings for further processing.

### Implementation of CLIP-Based Feature Extraction

The following Python Code extracts both image and text embeddings from the dataset using CLIP:

```
def extract_embeddings(self):
    """
    Extracts both text and image embeddings from the dataset.
    :return: Lists of image embeddings, text embeddings, filenames.
    """
    dataloader = DataLoader(self.dataset, batch_size=self.batch_size,
        shuffle=False)
    image_embeddings, text_embeddings, filenames, base_text = [], [],
        [], []

    with torch.no_grad():
        for batch in dataloader:
            images, batch_filenames, batch_types, batch_labels =
               batch

            # Convert images to CLIP format
            images = images.to(self.device)

            # Generate text descriptions dynamically
            text_descriptions = [self.generate_text_description(label
                , ctype)
                                 for ctype, label in zip(
                                    batch_types, batch_labels)]

            # Compute text and image embeddings
            text_tokens = clip.tokenize(text_descriptions).to(self.
               device)
            text_emb = self.model.encode_text(text_tokens)
            image_emb = self.model.encode_image(images)

            # Store embeddings and filenames
            image_embeddings.extend(image_emb.cpu().numpy())
            text_embeddings.extend(text_emb.cpu().numpy())
            filenames.extend(batch_filenames)
            base_text.extend(text_descriptions)

    return image_embeddings, text_embeddings, filenames, base_text
```

This implementation ensures that both image and text embeddings are computed efficiently in batches.

With CLIP embeddings extracted, the next step is to **fuse** them using a **Multilayer Perceptron (MLP)**. The fusion process is covered in the subsequent section.

### 3.2.5 Fusion Using an MLP

After extracting separate embeddings for images and text using CLIP, the next step is to combine them into a unified multimodal representation. We achieve this by using a **Multilayer Perceptron (MLP)**, which learns to integrate the information from both modalities into a **single feature space**. This fusion enables improved similarity computations, as it allows the model to capture both visual and textual characteristics in a shared representation.

#### What is a Multilayer Perceptron (MLP)?

A **Multilayer Perceptron (MLP)** is a type of artificial neural network that consists of multiple layers of fully connected neurons. Each neuron applies a weighted sum of its inputs, followed by a non-linear activation function (Goodfellow, Bengio, and Courville 2016). The fundamental components of an MLP are:

- **Input Layer:** Receives the input features (in our case, concatenated image and text embeddings).
- **Hidden Layers:** Transform the input using linear transformations followed by non-linear activation functions (e.g., ReLU).
- **Output Layer:** Produces the final fused embedding, representing both image and text features.

In our model, the input to the MLP is a concatenation of the CLIP image and text embeddings. The network learns a compressed feature representation that effectively combines information from both modalities.

#### Architecture of the Fusion Network

The MLP fusion network takes 512-dimensional image embeddings and 512-dimensional text embeddings, concatenates them into a 1024-dimensional vector, and processes them through two fully connected layers:

- **First Layer (Fully Connected)**: Projects the 1024-dimensional input into a lower-dimensional 256-dimensional space.
- **Activation Function (ReLU)**: Applies non-linearity to enable the network to learn complex relationships.
- **Second Layer (Fully Connected)**: Maps the 256-dimensional hidden representation to a final 128-dimensional multimodal embedding.

The final 128-dimensional fused embedding is used for similarity computations and further analysis.

```python
import torch
import torch.nn as nn

class MLPFusion(nn.Module):
    def __init__(self, input_dim=512, hidden_dim=256, output_dim=128):
        """
        MLP-based fusion of CLIP image and text embeddings.
        :param input_dim: Input feature dimension (512 for CLIP ViT-B/32).
        :param hidden_dim: Hidden layer size.
        :param output_dim: Output embedding size after fusion.
        """
        super(MLPFusion, self).__init__()

        self.fc1 = nn.Linear(input_dim * 2, hidden_dim)  # Concatenate img + text embeddings
        self.relu = nn.ReLU()
        self.fc2 = nn.Linear(hidden_dim, output_dim)  # Project to shared space

    def forward(self, img_emb, txt_emb):
        """
        Forward pass to fuse image and text embeddings.
        :param img_emb: Image embedding tensor (batch_size, 512).
        :param txt_emb: Text embedding tensor (batch_size, 512).
        :return: Fused embedding (batch_size, output_dim).
        """
        x = torch.cat((img_emb, txt_emb), dim=1)  # Concatenate embeddings
        x = self.relu(self.fc1(x))
        x = self.fc2(x)
        return x  # Return fused representation
```

### Loss Function for Multimodal Fusion

To train the fusion model effectively, we use Cosine Embedding Loss. This loss function ensures that similar image-text pairs are pulled closer together, while dissimilar pairs are pushed further apart in the embedding space.

**Cosine Similarity** measures how similar two vectors are, based on the angle between them:

$$\text{Cosine Similarity} = \frac{A \cdot B}{||A||||B||}$$

where $A$ and $B$ are the fused embeddings of the image and text.

The Cosine Embedding Loss is defined as:

$$L = \begin{cases} 1 - \cos(A, B), & \text{if labels are similar} \\ \max(0, \cos(A, B)), & \text{if labels are dissimilar} \end{cases}$$

This ensures that embeddings from related images and texts are aligned closely in feature space (Hadsell, Chopra, and LeCun 2006).

```
class FusionLoss(nn.Module):
    def __init__(self):
        super(FusionLoss, self).__init__()
        self.cosine_loss = nn.CosineEmbeddingLoss()

    def forward(self, fused_img, fused_txt, labels):
        """
        Compute contrastive loss for multimodal fusion.
        :param fused_img: Fused image embeddings.
        :param fused_txt: Fused text embeddings.
        :param labels: 1 for matching pairs, -1 for non-matching
   pairs.
        """
        return self.cosine_loss(fused_img, fused_txt, labels)
```

#### Final Fused Embedding Dimension

The final output of the fusion model is a 128-dimensional feature vector. This compressed representation is a rich multimodal encoding that integrates both visual and textual information about the coin.

- The **image** contributes information about the coin's shape, texture, and engravings.
- The **text** adds semantic meaning related to coin history and category.

By merging these modalities, we improve similarity computations and provide a more robust feature space for coin classification.

### 3.2.6 t-SNE Visualization of Fused Features

In order to gain insights into the effectiveness of our multimodal fusion, we apply **t-Distributed Stochastic Neighbor Embedding (t-SNE)** to the fused embeddings (Maaten and G. Hinton 2008). This allows us to visually assess whether the fused embeddings exhibit meaningful clusters, which would indicate that the model is learning useful representations of the coins.

Figure 3.6 and 3.7 show the t-SNE projection of the fused image-text embeddings for both obverse and reverse coin sides.

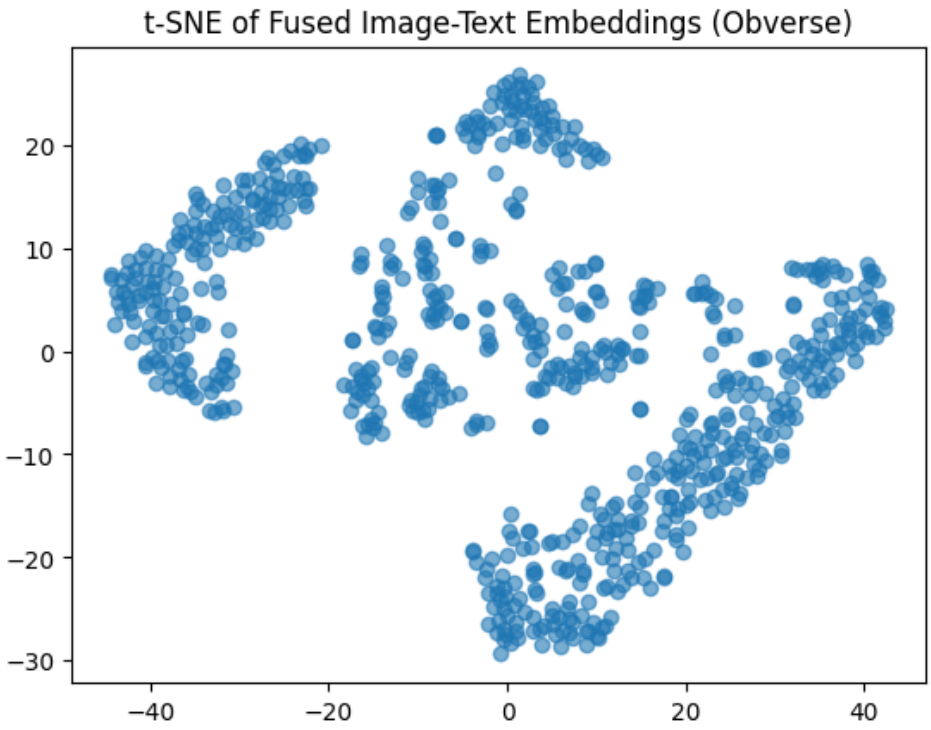


Figure 3.6: t-SNE Visualization of Fused Image-Text Embeddings for Obverse Coins.

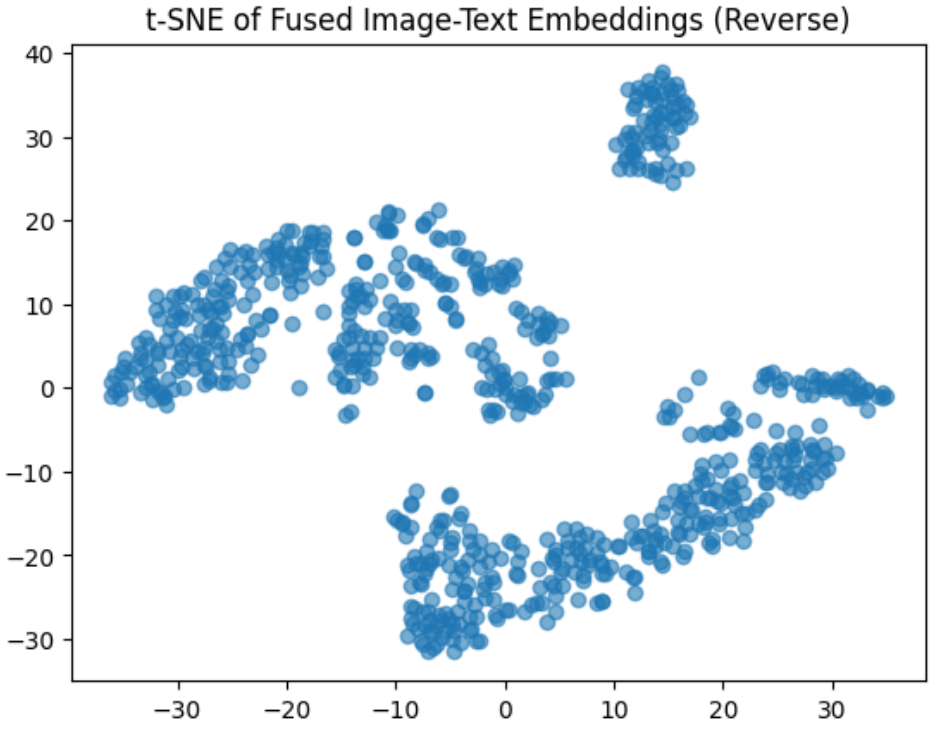


Figure 3.7: t-SNE Visualization of Fused Image-Text Embeddings for Reverse Coins.

#### Observations

From the visualizations, we can make the following observations:

- **Formation of Clusters**: Both the obverse and reverse embeddings show distinct clusters, suggesting that the multimodal fusion process successfully captures meaningful relationships between coin images and their textual descriptions.
- **Separation of Coin Types**: The presence of well-defined groups indicates that similar coin types tend to be mapped close together in the fused feature space. This is a strong indication that the fusion model is effectively aligning the visual and textual modalities.
- **Potential Outliers**: In both plots, there are a few data points that are noticeably distant from the main clusters. These might correspond to coins that lack clear textual descriptions or have unique features that distinguish them from the majority.

These results confirm that our approach to multimodal fusion preserves meaningful relationships between images and text while structuring the latent space in a way that allows for intuitive organization of different coin types.

### 3.2.7 Summary of Multimodal Feature Extraction

In this section, we introduced a multimodal approach to feature extraction by combining both image and textual information using CLIP embeddings. We began by discussing the dataset processing steps, where coin images were categorized based on their types, and how textual descriptions were generated dynamically using structured templates. This allowed us to create meaningful text embeddings that align well with the visual representations.

We then explored the methodology behind CLIP and how it enables joint vision-language learning, followed by the extraction of image and text features. To enhance the multimodal representation, we fused these embeddings using a **Multi-Layer Perceptron (MLP)**, effectively combining visual and textual information into a unified latent space. The fusion process was guided by a **Cosine Embedding Loss**, ensuring that corresponding image-text pairs remain close in the embedding space.

To evaluate the quality of the fused embeddings, we visualized them using **t-SNE**, which demonstrated meaningful clustering patterns, indicating that the fusion model successfully captured structural relationships between different coin types.

Our findings suggest that multimodal feature extraction enhances the ability to learn meaningful representations of ancient coins. This approach lays the foundation for further comparisons in the next section, where we analyze the differences in similarity computations between image-only embeddings and fused image-text embeddings.

## 3.3 Computing Similarity Across Embeddings

### 3.3.1 Introduction to Cosine Similarity

Cosine similarity is a widely used metric for measuring the similarity between two high-dimensional vectors. Given two vectors $A$ and $B$, the cosine similarity is computed as:

$$\text{cosine\_sim}(A, B) = \frac{A \cdot B}{\|A\|\|B\|} \tag{3.3}$$

where $A \cdot B$ represents the dot product of the two vectors, and $\|A\|$ and $\|B\|$ denote their Euclidean norms. This metric is particularly useful for our study because it measures the orientation between vectors rather than their magnitude, making it well-suited for comparing feature representations in an embedding space.

In the context of our study, we use cosine similarity to measure the similarity between:

- **Image-based embeddings:** Extracted using a pretrained ResNet50 model.
- **Multimodal embeddings:** Obtained by fusing CLIP image and text representations through an MLP.

By comparing these embeddings, we can analyze how the fusion of textual and visual information impacts similarity retrieval.

### 3.3.2 Measuring Similarity for Image-Based Embeddings

For image-based embeddings, we use the feature vectors extracted from a pretrained ResNet50 model. These embeddings are stored in `.npy` files for both obverse and reverse sides of the coins.

The similarity retrieval process follows these steps:

1. Load the stored embeddings and their corresponding filenames.
2. Compute the cosine similarity between a query image and all other images in the dataset.
3. Retrieve the most and least similar images based on similarity scores.

This process allows us to analyze how well ResNet50 features capture the **visual similarity** between coins. The retrieved images demonstrate how the network distinguishes between similar and dissimilar coin patterns.

### 3.3.3 Measuring Similarity in Multimodal Space

In contrast to image-based similarity, the multimodal approach integrates **both visual and textual features** to create a more enriched representation. We utilize CLIP embeddings, where:

- The image embeddings are extracted from the CLIP vision encoder.
- The text embeddings are derived from templates describing the coin's characteristics.
- These two embeddings are fused using an MLP to produce a final multimodal representation.

The similarity retrieval process remains the same: cosine similarity is computed between the fused representations of a query coin and all others in the dataset. This method allows us to leverage textual context, potentially improving retrieval performance by aligning visual features with descriptive labels.

### 3.3.4 Comparison of Results

After computing similarity scores for both embedding types, we analyze the differences between the two approaches. Below are some examples illustrating the retrieved images for a given query:


Figure 3.8: Example 1: Similarity retrieval using fused embeddings (text+image).

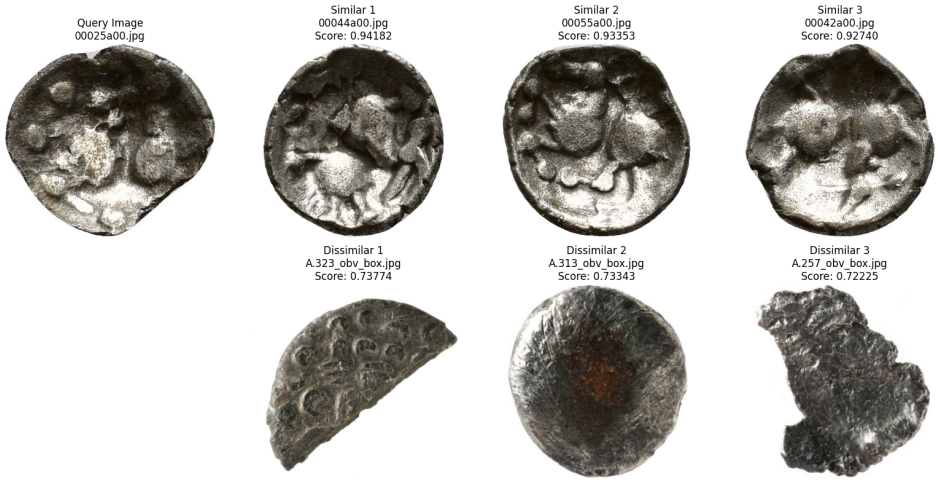


Figure 3.9: Example 1: Similarity retrieval using image-only embeddings.


Figure 3.10: Example 2: Similarity retrieval using fused embeddings (text+image).


Figure 3.11: Example 2: Similarity retrieval using image-only embeddings.


Figure 3.12: Example 3: Similarity retrieval using fused embeddings (text+image).


Figure 3.13: Example 3: Similarity retrieval using image-only embeddings.

As observed, the **multimodal approach produces high similarity scores across all images**, sometimes making it difficult to differentiate between similar and dissimilar coins. Meanwhile, the **image-based approach exhibits a more diverse range of similarity values**, providing clearer separation between different coin types.

This analysis highlights the impact of incorporating textual information into the embedding space and its effects on similarity computations.

## 3.4 Network Analysis of Coin Similarities

### 3.4.1 Introduction

In this section, we analyze the relationships between coin embeddings by constructing a similarity network. Each node represents a coin, and edges are drawn between coins whose

cosine similarity exceeds a defined threshold. This approach helps us understand how coins are clustered based on their visual and multimodal embeddings.

### 3.4.2 Building the Similarity Network

To construct the network, we used the precomputed embeddings from both the **ResNet50-based image features** and the **fused CLIP embeddings**. The similarity between two nodes (coins) is determined using cosine similarity. If the similarity is above a predefined threshold, an edge is drawn between them.

A major challenge was selecting an appropriate threshold. For image-based embeddings, a reasonable threshold (e.g., 0.95) provided meaningful relationships. However, for fused embeddings, the similarity values were significantly higher, often exceeding 0.99, making differentiation between coins more difficult.

### 3.4.3 Visualization of Similarity Graphs

To better interpret the similarity relationships, we generated interactive network graphs. These graphs were uploaded to GitHub Pages, allowing for dynamic exploration. The following links provide access to the interactive visualizations:

- https://kenankhauto.github.io/AI-driven-coin-analysis/obverse_fused_graph_0.9995.html
- https://kenankhauto.github.io/AI-driven-coin-analysis/reverse_fused_graph_0.9995.html
- https://kenankhauto.github.io/AI-driven-coin-analysis/obverse_img_graph_0.95.html
- https://kenankhauto.github.io/AI-driven-coin-analysis/reverse_img_graph_0.95.html

In Figure 3.14, we provide an example of a static visualization of the similarity graph.

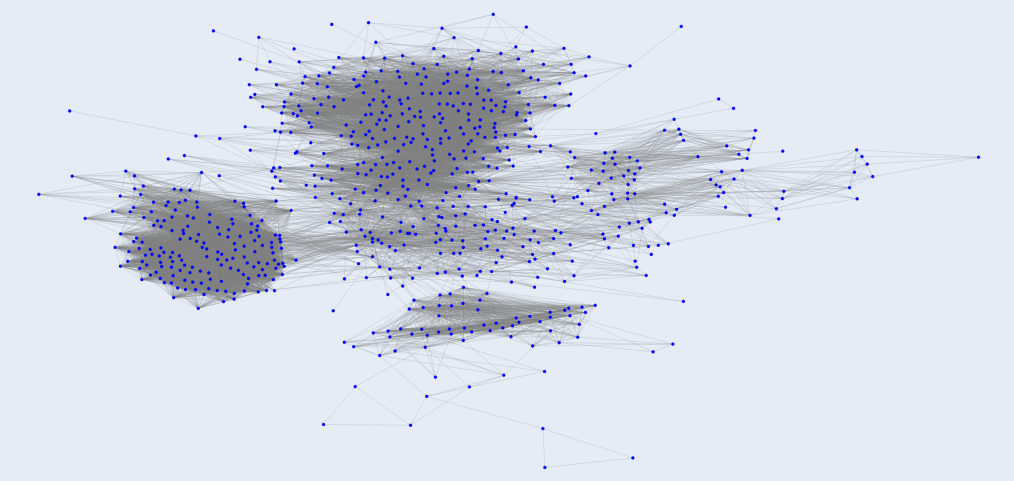

Figure 3.14: Example of a similarity network for obverse fused embeddings (0.9995).

### 3.4.4 Observations

From the network visualizations, we observe the following:

- The **image-based embeddings** form more distinct clusters, with noticeable separation between different coin types.
- The **fused embeddings** form a denser network, with almost all nodes being interconnected due to the high similarity scores.
- Some outlier coins exist, which do not strongly connect to any group.

### 3.4.5 Limitations and Challenges

Despite the insights gained from the network, there are some challenges:

- **Threshold Sensitivity:** A small change in the similarity threshold significantly alters the network structure.
- **High Similarity in Fused Embeddings:** The fused features tend to have artificially high similarity values, making fine-grained distinctions difficult.
- **Lack of Ground Truth:** Since we lack labeled relationships between coins, it is difficult to validate the network against an established classification.

### 3.4.6 Conclusion

The network analysis provided valuable insights into the structure of our coin embeddings. The image-based embeddings formed distinct clusters, while the fused embeddings created a more interconnected network. These results suggest that multimodal fusion enhances general similarity capture but may require additional fine-tuning to differentiate similar yet distinct coins.

# 4 Evaluation

Evaluating the performance of the proposed feature extraction and similarity retrieval methods is essential to assess their effectiveness in distinguishing and grouping numismatic objects. The evaluation aims to analyze the quality of the extracted features, the impact of multimodal fusion on similarity retrieval, and the overall performance in retrieving semantically relevant coins.

This chapter is structured as follows:

- In Section 4.1, we revisit the *t-SNE visualization* of extracted embeddings to observe clustering behaviors and analyze the feature space.
- Section 4.2 provides a quantitative analysis of *similarity retrieval*, comparing the performance of image-only embeddings with multimodal fused embeddings.
- Section 4.3 discusses the main issues we observed, and potential solutions.
- Finally, Section 4.4 summarizes key findings and discusses their implications for numismatic image analysis.

The primary goal of this evaluation is to determine whether the fusion of image and text embeddings improves retrieval performance and whether the extracted features can effectively cluster similar coin types while maintaining meaningful separability among different categories.

## 4.1 t-SNE Feature Space Evaluation

To evaluate the structure of the extracted feature space, we employ *t-Distributed Stochastic Neighbor Embedding* (t-SNE) Maaten and G. Hinton 2008, a widely used dimensionality reduction technique for visualizing high-dimensional data. The t-SNE plots help us understand whether the extracted features exhibit meaningful clustering and separability.

### 4.1.1 Analysis of Image-Based Embeddings

In previous sections, we generated t-SNE plots for the embeddings extracted using a pretrained ResNet-50 model. As shown in Figures 3.3 and 3.4, the distribution of image-based features does not exhibit clear, well-defined clusters. While some local grouping is observable, there is a significant overlap between different coin types, indicating that image-based embeddings alone are not highly discriminative for certain coin classes.

This lack of strong clustering suggests that visual features alone may not be sufficient to capture the semantic differences between numismatic objects. In cases where coins have similar visual characteristics, the embeddings may fail to differentiate between distinct categories.

### 4.1.2 Analysis of Fused Image-Text Embeddings

In contrast, the t-SNE plots for the fused image-text embeddings, displayed in Figures 3.6 and 3.7, show a remarkable improvement in clustering quality. Compared to the image-only features, these embeddings form well-separated and compact clusters, indicating that incorporating textual descriptions significantly enhances the feature space's semantic organization.

The improved clustering demonstrates that the multimodal fusion successfully leverages textual descriptions to refine feature representation. Coins with similar textual annotations tend to be placed closer in the embedding space, leading to a more meaningful grouping. This highlights the advantage of combining both visual and textual information to improve coin classification and retrieval tasks.

### 4.1.3 Key Observations

Based on the visual analysis of the t-SNE projections, the following key insights can be drawn:

- **Image-based embeddings** show moderate clustering tendencies but lack strong separation, leading to ambiguous feature representations.
- **Fused multimodal embeddings** produce well-defined and compact clusters, demonstrating improved feature discrimination.
- The incorporation of textual descriptions reduces feature overlap and strengthens the semantic meaning of embeddings, making them more robust for similarity-based retrieval.

These findings highlight the importance of leveraging textual information in conjunction with visual features to enhance the understanding and classification of numismatic objects.

## 4.2 Similarity Retrieval Performance

In this section, we evaluate the similarity retrieval performance of both **image-only embeddings** and **fused text-image embeddings**. The goal of this analysis is to determine how well each embedding method retrieves semantically similar coins and whether the inclusion of textual features improves the retrieval process.

### 4.2.1 Retrieval Using Image-Only Embeddings

For image-based retrieval, we computed cosine similarity between the extracted features using the ResNet-50 model. The top-3 most similar and top-3 least similar images were retrieved based on similarity scores.

As seen in Figure 3.9, Figure 3.11, and Figure 3.13, the image-only embeddings provided reasonable similarity retrieval results. The retrieved images often shared strong **visual similarities** with the query image, such as texture, and overall structure.

However, one key observation is that this method struggled to recognize **different stylistic variations of the same coin type**. For example, heavily worn coins of the same type appeared visually different and were not always retrieved as the most similar images. Additionally, the dissimilar images had noticeably lower similarity scores, demonstrating that the model effectively separated different classes.

### 4.2.2 Retrieval Using Fused Image-Text Embeddings

By incorporating textual descriptions into the embeddings, we aimed to enhance the retrieval process by introducing **semantic understanding** beyond pixel-level similarity. The retrieval was performed using multimodal embeddings obtained from the MLP fusion of CLIP text and image features.

As shown in Figure 3.8, Figure 3.10, and Figure 3.12, the fused embeddings successfully retrieved images that were semantically closer based on the textual descriptions. This allowed the retrieval of coins that shared common symbols and historical attributes, even if they had noticeable visual differences.

However, one drawback was observed: **similarity scores were consistently high, even for dissimilar images**. This suggests that while the textual information contributed useful context, it also introduced noise, possibly due to the template-based text descriptions. As a result, even when retrieving highly dissimilar coins, the similarity scores remained above a certain threshold, reducing the relative contrast between similar and dissimilar items.

### 4.2.3 Comparison of Both Approaches

The comparison between both retrieval methods reveals key trade-offs:

- **Image-only embeddings** produced more reliable visual similarity scores and effectively separated different classes but struggled with stylistic variations of the same type.
- **Fused embeddings** provided better *semantic* retrieval, identifying coins of the same type even when visually different, but suffered from high similarity scores even for dissimilar items.

These observations suggest that while multimodal fusion enhances contextual understanding, further refinement is needed to ensure that the retrieved similarity scores reflect true semantic distances.

## 4.3 Limitations and Error Analysis

While the proposed multimodal embedding approach significantly enhances similarity retrieval by integrating textual and visual features, it also introduces several limitations that must be addressed. This section discusses the key challenges encountered during the study.

### 4.3.1 Influence of Template-Based Text Embeddings

One of the primary limitations of the fused embeddings is the reliance on **predefined text templates** for generating textual descriptions. Although these templates ensure a structured format for textual input, they may inadvertently introduce bias into the embeddings.

Since all textual descriptions follow a similar structure (e.g., "This is the obverse of an ancient Celtic coin depicting [Type]"), the model may learn to associate common textual patterns rather than actual semantic differences. This can lead to an artificial increase in similarity scores for coins that share generic words in their descriptions, even if their visual features are distinct.

**Potential Solution:** Future work could explore dynamic text descriptions by integrating additional metadata (e.g., historical context, minting techniques) or employing generative models to produce richer textual descriptions.

### 4.3.2 High Similarity Scores for Dissimilar Images

As observed in Section 4.2, the similarity scores for multimodal embeddings tend to be consistently high, even for visually dissimilar images. This is particularly evident when comparing the least similar images retrieved using image-only embeddings (which show a clear drop in similarity scores) versus the fused embeddings, where the score differences are minimal.

**Possible Causes:**

- CLIP embeddings naturally encode semantic meaning, which may result in over-smoothing similarity values.
- The text embeddings contribute additional information but may overpower fine-grained visual differences.
- The concatenation-based fusion method may not fully capture complementary details from both modalities.

**Potential Solution:** Instead of directly concatenating features, alternative fusion techniques such as *attention-based mechanisms* or *modality weighting* could be explored to ensure that the image and text contributions are balanced.

### 4.3.3 Challenges in Handling Unclassified Coins

A subset of the dataset consists of images that lack a predefined type classification. Since these images do not have a textual description, they do not benefit from the advantages of multimodal fusion and rely solely on visual similarity.

The absence of textual context can lead to incorrect retrieval results when using fused embeddings, as these images are mapped differently in the embedding space. This issue highlights the need for a fallback mechanism when text information is unavailable.

**Potential Solution:** A hybrid retrieval system could be implemented, where images without textual descriptions default to image-only similarity computations while retaining the option to incorporate text if classifications become available in the future.

### 4.3.4 Limitations of the Current Evaluation Metrics

The evaluation relied on **cosine similarity** as the primary metric for retrieval performance. While this method is widely used in embedding-based similarity tasks, it has certain drawbacks:

- It does not directly account for **class separability** in the feature space.
- Similarity scores do not necessarily correspond to perceptual similarity.
- A lack of ground truth labels for all images makes it difficult to compute traditional classification metrics such as precision and recall.

**Potential Solution:** Future work could integrate clustering-based evaluation metrics (e.g., *Silhouette Score, Davies-Bouldin Index*) to assess how well the embeddings separate distinct types. Additionally, collecting expert-labeled similarity data would enable a more rigorous quantitative evaluation.

### 4.3.5 Impact of Image Preprocessing Choices

Finally, image preprocessing steps such as grayscale conversion and normalization influence feature extraction and similarity computations. Since CLIP was originally trained on full-color images, converting coins to grayscale may have caused a loss of discriminative color-based features.

**Potential Solution:** A comparative study could be conducted to evaluate performance differences between color and grayscale embeddings, ensuring that preprocessing choices align with the optimal use of CLIP.

### 4.3.6 Summary of Limitations and Future Directions

To summarize, the main limitations identified in this study include:

- Bias introduced by template-based text descriptions.
- High similarity scores, even for visually distinct images.
- Challenges in handling unclassified coins.
- Constraints of cosine similarity as a retrieval metric.
- Potential loss of information due to image preprocessing.

Future research should explore advanced fusion techniques, alternative evaluation methods, and dynamic text embeddings to address these challenges and further improve the accuracy of multimodal coin similarity retrieval.

## 4.4 Summary of the Evaluation

This chapter evaluated the effectiveness of different embedding approaches for coin similarity retrieval. We analyzed both **image-based** and **multimodal fused** embeddings, comparing their ability to cluster similar coins and retrieve relevant matches.

**Key observations:**

- **t-SNE visualizations** (3.6, 3.7) demonstrated that fused embeddings exhibit clearer clustering compared to image-only embeddings, indicating better semantic separation.
- While fused embeddings improved retrieval of relevant matches, they also exhibited **overly high similarity scores**, even for dissimilar images (3.8, 3.10).
- Template-based text embeddings may have contributed to similarity distortions, as discussed in Section 4.3.1.
- Cosine similarity proved useful for comparing embeddings, but alternative metrics or additional clustering techniques could further enhance evaluation.

Despite the limitations identified in Section 4.3, the overall results suggest that multimodal embeddings can significantly improve numismatic research by enabling automated similarity retrieval. However, further refinement of text templates and fusion strategies is necessary to reduce similarity distortions.

The next chapter will provide a **final conclusion** and discuss possible directions for *future research*, including methods for improving embedding quality and incorporating additional metadata into the analysis.

# 5 Conclusion and Future Work

## 5.1 Summary of Contributions

This study explored the application of **multimodal embeddings** for numismatic coin similarity retrieval. We proposed a novel approach that combines **image and text embeddings** using CLIP and an **MLP-based fusion network**, allowing for improved similarity detection compared to image-only methods.

The key contributions of this thesis include:

- A novel dataset processing pipeline that extracts coin types from structured folders.
- The use of a Pretrained-Resnet50 for visual coin representation.
- The use of CLIP embeddings for both textual and visual coin representation.
- A custom template-based text embedding strategy to describe numismatic images.
- The fusion of text and image features via an MLP network, enhancing retrieval.
- The use of network analysis techniques to investigate structural relationships between coin similarities.
- A detailed evaluation comparing similarity retrieval across different embedding types.

## 5.2 Key Findings

Our experiments led to the following conclusions:

- The fused embeddings provided better clustering in **t-SNE visualizations** (3.6, 3.7), showing that text information contributes valuable semantic meaning.
- Image-based embeddings alone struggled to form clear clusters, indicating that visual features alone may not be sufficient for nuanced coin classification.
- Similarity retrieval using fused embeddings performed well but sometimes exhibited overly high similarity scores, even for visually distinct images (3.8, 3.9).
- Text templates played a crucial role but may have introduced **bias**, which impacted the similarity results (4.3.1).

These findings suggest that multimodal fusion enhances similarity retrieval in numismatics but requires refinement to handle textual biases and over-similarity.

## 5.3 Limitations of the Study

While this study presents significant advancements, there are still limitations:

- **Limited dataset size**: The study was conducted on a specific numismatic dataset. Testing on a larger dataset could provide more robust conclusions.
- **Textual bias from templates**: The predefined templates may not always reflect the exact historical or artistic significance of each coin.
- **High similarity scores**: The fused embeddings tend to produce very high similarity scores even for dissimilar images, potentially reducing retrieval effectiveness.
- **Lack of ground truth labels**: Due to the absence of exact similarity ground truth, evaluation relied primarily on visualization and relative similarity measures.

## 5.4 Future Work

To address these limitations, several future directions could be explored:

### 5.4.1 Dynamic Text Embeddings

Instead of using fixed templates, integrating **Natural Language Processing (NLP)** techniques such as **large language models (LLMs)** could generate more contextually rich descriptions, improving text embeddings.

### 5.4.2 Advanced Fusion Strategies

The MLP-based fusion method could be replaced or extended with:

- **Attention-based fusion** mechanisms to better weight important modalities.
- **Graph Neural Networks (GNNs)** to model relationships between coin images more effectively.

### 5.4.3 Alternative Similarity Metrics

While cosine similarity was used in this study, future work could evaluate:

- Mahalanobis distance for similarity retrieval.
- Learned similarity functions via contrastive learning.

### 5.4.4 Incorporating Additional Metadata

Additional metadata such as:

- Minting date and location,
- Material composition,
- Numismatic provenance,

could be integrated to provide a richer multimodal analysis.

### 5.4.5 Expanding Network Analysis

The graph-based approach for similarity networks could be improved by:

- Using community detection algorithms to classify coins into groups.
- Applying PageRank or centrality measures to detect historically significant coins.

## 5.5 Final Remarks

This thesis demonstrated the power of **multimodal learning** in numismatic research. By combining image and text embeddings, we showed that similarity retrieval can be improved significantly. However, the results also highlight the need for better text representations, alternative fusion strategies, and improved evaluation metrics.

The findings of this study provide a foundation for further research into automated numismatic analysis, with potential applications in **museum collections, archaeological studies, and digital heritage preservation**.

Future work should focus on refining the fusion process, integrating richer metadata, and exploring new similarity measurement techniques to enhance the precision and usability of AI-driven coin classification systems.

# Bibliography


Bishop, Christopher M. (2006). *Pattern recognition and machine learning*. Springer.

Canny, John (1986). "A Computational Approach to Edge Detection". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-8.6, pp. 679–698. DOI: 10.1109/TPAMI.1986.4767851.

Chen, Ting, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton (2020). "A Simple Framework for Contrastive Learning of Visual Representations". In: *International Conference on Machine Learning (ICML)*. DOI: 10.48550/arXiv.2002.05709.

Chen, Xinlei, Saining Xie, and Kaiming He (2021). "An Empirical Study of Training Self-Supervised Vision Transformers". In: *International Conference on Computer Vision (ICCV)*. DOI: 10.48550/arXiv.2104.02057.

CoinsWeekly (2023). "How AI Is Transforming Numismatics". In: *CoinsWeekly*. URL: https://new.coinsweekly.com/news-en/how-ai-is-transforming-numismatics/.

Dalal, Navneet and Bill Triggs (2005). "Histograms of Oriented Gradients for Human Detection". In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. Vol. 1, pp. 886–893. DOI: 10.1109/CVPR.2005.177.

Deng, Jia et al. (2009). "ImageNet: A large-scale hierarchical image database". In: *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pp. 248–255.

Dosovitskiy, Alexey et al. (2021). "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". In: *International Conference on Learning Representations (ICLR)*. DOI: 10.48550/arXiv.2010.11929.

Exchange, The Old Currency (2016). "Difficulties in Grading Old Coins". In: *The Old Currency Exchange*. URL: https://oldcurrencyexchange.com/2016/03/28/difficulties-in-grading-old-coins/.

Gonzalez, Rafael C. and Richard E. Woods (2002). *Digital Image Processing*. 2nd. Prentice Hall.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016). *Deep learning*. MIT press.

Grill, Jean-Bastien et al. (2020). "Bootstrap Your Own Latent: A New Approach to Self-Supervised Learning". In: *Advances in Neural Information Processing Systems (NeurIPS)*. DOI: 10.48550/arXiv.2006.07733.

Hadsell, Raia, Sumit Chopra, and Yann LeCun (2006). "Dimensionality reduction by learning an invariant mapping". In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*. Vol. 2. IEEE, pp. 1735–1742.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (2016). "Deep Residual Learning for Image Recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778. DOI: 10.1109/CVPR.2016.90.

Hinh, Justin (2023). "Numi: AI-Powered Coin Grading and Identification". In: *Justin Hinh's Portfolio*. URL: https://justinhinh.webflow.io/projects/numi.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2012). "ImageNet classification with deep convolutional neural networks". In: *Advances in neural information processing systems* 25, pp. 1097–1105.

LeCun, Yann, Léon Bottou, Yoshua Bengio, and Patrick Haffner (1998). "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11, pp. 2278–2324. DOI: 10.1109/5.726791.

Lowe, David G. (2004). "Distinctive Image Features from Scale-Invariant Keypoints". In: *International Journal of Computer Vision* 60.2, pp. 91–110. DOI: 10.1023/B:VISI.0000029664.99615.94.

Maaten, Laurens van der and Geoffrey Hinton (2008). "Visualizing data using t-SNE". In: *Journal of Machine Learning Research* 9, pp. 2579–2605.

News, Numismatic (2023a). "Biases In Certified Coin Populations". In: *Numismatic News*. URL: https://www.numismaticnews.net/coin-market/biases-in-certified-coin-populations.

— (2023b). "The Challenge of Grading Circulating Coins". In: *Numismatic News*. URL: https://www.numismaticnews.net/collecting-101/the-challenge-of-grading-circulating-coins.

Pedregosa, Fabian et al. (2011). *Scikit-learn: Machine Learning in Python*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html.

Radford, Alec et al. (2021). "Learning Transferable Visual Models From Natural Language Supervision". In: *arXiv preprint arXiv:2103.00020*. URL: https://arxiv.org/abs/2103.00020.

Simonyan, Karen and Andrew Zisserman (2015). "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *International Conference on Learning Representations (ICLR)*. DOI: 10.48550/arXiv.1409.1556.

Tan, Mingxing and Quoc V. Le (2019). "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks". In: *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 6105–6114. DOI: 10.48550/arXiv.1905.11946.